What is claimed is:

1. A logic verification system, comprising:

a hardware simulator;

a diagnostic system;

an interprocess communication mechanism for transferring stimulus from the diagnostic system to the hardware simulator and for transferring results from the hardware simulator to the diagnostic system;

a verification environment, wherein the verification environment includes two or more layers of abstraction placed between a device being verified and a diagnostic program, wherein one of the layers of abstraction is a verification kernel, wherein the verification kernel includes a diagnostic kernel and a simulation kernel and wherein the diagnostic kernel and the simulation communicate through the interprocess communication mechanism.

2. The logic verification system according to claim 1, wherein the diagnostic system includes a diagnostic program, wherein the verification environment further includes a diagnostic programming interface, and wherein the diagnostic program, the diagnostic programming interface and the diagnostic kernel are linked to form a diagnostic test.

3. The logic verification system according to claim 2, wherein the diagnostic programming interface is a program library that adapts abstract data structures in the diagnostic program to structures in the verification kernel.

4. The logic verification system according to claim 2, wherein the diagnostic kernel is a program library which operates on events.

5. The logic verification system according to claim 2, wherein the diagnostic kernel is a program library which operates on wiggles and bundles.

6. The logic verification system according to claim 2, wherein the verification environment further includes a logic design representative of the device being verified and a logic design wrapper, and wherein the logic design, the logic design wrapper and the simulation kernel are linked to form a logic simulator.

7. The logic verification system according to claim 6, wherein the logic design wrapper is a program library that adapts the simulation kernel to the logic design.

8. The logic verification system according to claim 6, wherein the simulation kernel is a program library which operates on events.

9. The logic verification system according to claim 6, wherein the simulation kernel is a program library which operates on wiggles and bundles.

10. The logic verification system according to claim 2, wherein the verification environment further includes a logic design representative of the device being verified and a logic design wrapper, and wherein the logic design, the logic design wrapper and the simulation kernel are linked to form a logic simulator.

11. The logic verification system according to claim 10, wherein the logic design wrapper is a program library that adapts the simulation kernel to the logic design.

12. The logic verification system according to claim 10, wherein the simulation kernel is a program library which operates on events.

13. The logic verification system according to claim 10, wherein the simulation kernel is a program library which operates on wiggles and bundles.

14. The logic verification system according to claim 1, wherein the verification environment further includes a group data structure used to declare an array of undetermined length.

15. The logic verification system according to claim 14, wherein the verification environment further includes a length function, wherein the length function returns length of an array declared using the group data structure.

16. The logic verification system according to claim 1, wherein the verification environment further includes a four-state variable data structure.

17. The logic verification system according to claim 1, wherein the verification environment further includes a plurality of ports, including an event port, wherein the event port drives and captures events.

18. The logic verification system according to claim 17, wherein the plurality of ports includes a memory back door port, wherein the memory back door port is used to construct a memory model.

19. The logic verification system according to claim 1, wherein the verification environment diagnostic program generates stimulus via a DPI apply.

20. A method of verifying an electronic system, comprising:

providing a verification kernel;

expressing the electronic system as a logic design;

defining a wrapper, wherein the wrapper is an interface between the logic design and the verification kernel;

placing tests to be run against the logic design within a diagnostic program;

defining a diagnostic program interface, where the diagnostic program interface is an interface between the diagnostic program and the verification kernel;

executing the tests against the logic design;

reporting results of the tests; and

validating the results against expected results.

21. The method according to claim 20, wherein defining a wrapper includes compiling the logic design and the wrapper into an object file, wherein the object file is linked with verification kernel routines to create an executable logic simulator.

22. The method according to claim 21, wherein executing the tests against the logic design includes establishing an inter-process communication (IPC) layer between the diagnostic program and the logic simulator.

23. The method according to claim 20 wherein executing includes trapping to a handler routine on occurrence of an exception.

24. The method according to claim 20 wherein executing includes executing recovery code when an exception is detected.

25. The method according to claim 20 wherein executing includes cooperative multitasking of a plurality of threads.

26. The method according to claim 25, wherein cooperative multitasking includes communicating between threads via a semaphore variable.

27. The method according to claim 25, wherein cooperative multitasking includes controlling execution of threads via a barrier function.

28. The method according to claim 20 wherein executing includes waiting for a nondeterministic outcome.

29. The method according to claim 20 wherein executing includes popping events off an event queue.

30. The method according to claim 20 wherein executing includes popping wiggles off a wiggle queue and popping bundles off one or more bundles queues.

31. The method according to claim 20 wherein executing includes referencing memories built using memory access PLI tasks.

32. The method according to claim 20 wherein executing includes performing an examine function to detect a change of state.

33. A system for simulating operation of an electronic device, comprising:

a hardware simulator;

a diagnostic system; and

an interprocess communication mechanism for transferring stimulus from the diagnostic system to the hardware simulator and for transferring results from the hardware simulator to the diagnostic system;

wherein the simulator is capable of receiving connections and commands from a diagnostic program running on the diagnostic system, of executing the commands as directed and of returning results to the diagnostic system.

34. The system according to claim 33, wherein the diagnostic system includes a simulator kernel, a logic design wrapper and a logic design representative of the electronic device.

35. The system according to claim 34, wherein the logic design wrapper is a program library that adapts the simulation kernel to the logic design.

36. The system according to claim 35, wherein the simulation kernel is a program library which operates on events.

37. The system according to claim 35, wherein the simulation kernel is a program library which operates on wiggles and bundles.

38. The system according to claim 33, wherein the diagnostic system includes a group data structure used to declare an array of undetermined length.

39. The system according to claim 33, wherein the diagnostic system includes a length function, wherein the length function returns length of an array declared using the group data structure.

40. The system according to claim 33, wherein the diagnostic system includes a four-state variable data structure.

41. The system according to claim 33, wherein the diagnostic system includes a plurality of ports, including an event port, wherein the event port drives and captures events.

42. The system according to claim 41, wherein the plurality of ports includes a memory back door port, wherein the memory back door port is used to construct a memory model.

43. The system according to claim 33, wherein the diagnostic program generates stimulus via a DPI apply.